

NAKI-AN73
MASATO SUZUKI et al.

本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されて
も事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
this Office.

願 年 月 日
Date of Application:

1993年 5月27日

願 番 号
Application Number:

平成 5年特許願第126212号

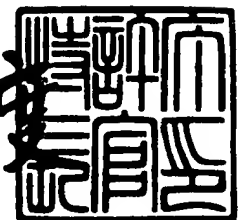
願 人
Applicant(s):

松下電器産業株式会社

1994年 3月11日

特許庁長官
Commissioner,
Patent Office

麻 生 渡



出証番号 出証特平06-3008641

【書類名】 特許願

【整理番号】 2030452090

【提出日】 平成 5年 5月27日

【あて先】 特許庁長官 麻 生 渡 殿

【国際特許分類】 G06F 9/44

【発明の名称】 プログラム変換装置およびプロセッサ

【請求項の数】 5

【発明者】

【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

【氏名】 鈴木 正人

【発明者】

【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

【氏名】 神山 祐史

【発明者】

【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

【氏名】 宮地 信哉

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代表者】 森下 洋一

【代理人】

【識別番号】 100090446

【弁理士】

【氏名又は名称】 中島 司朗

【手数料の表示】

【納付方法】 予納

【予納台帳番号】 014823

【納付金額】 14,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9003742

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム変換装置およびプロセッサ

【特許請求の範囲】

【請求項1】 主たるデータの型における有効なビット幅Mよりも長いビット幅Nをもつアドレスを扱うプロセッサを対象とし、高級言語プログラムに基づいて機械語命令を生成するプログラム変換装置であって、

主たるデータの型における有効なビット幅およびアドレスのビット幅を保持するパラメータ保持手段と、

生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成し、アドレスを表す変数である場合、パラメータ保持手段が保持するアドレスのビット幅を有効とする命令を生成する生成手段と

を備えたことを特徴とするプログラム変換装置。

【請求項2】 前記データのビット幅Mは16であり、

前記アドレスのビット幅Nは17以上31以下であることを特徴とする請求項1記載のプログラム変換装置。

【請求項3】 前記生成手段は、

生成すべき機械語命令がメモリをアクセスする命令である場合に、アクセス対象となる変数がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅をアクセス幅とすべきことを指示するメモリ管理手段と、

生成すべき機械語命令がレジスタを使用する命令である場合に、そのレジスタにリード又はライトされる変数がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅を有効とすべきことを指示するレジスタ管理手段と、

生成すべき機械語命令が即値を使用する命令である場合にその即値がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅の即値を用いるべきことを指示する即値管理手段と

メモリ管理手段、レジスタ管理手段及び即値管理手段からの指示に従って、機

械語命令を生成するコード生成手段と

からなることを特徴とする請求項1又は2記載のプログラム変換装置。

【請求項4】 請求項1ないし3記載のいずれかのプログラム変換装置により生成されたプログラムを実行するプロセッサであって、

Nビット長のプログラムカウンタと、

Nビット長の演算、及びデータのビット幅であるMビット長の演算を実行する実行手段と

を備えたことを特徴とするプロセッサ。

【請求項5】 前記Nは24である

ことを特徴とする請求項3記載のプロセッサ。

【発明の詳細な説明】

【0001】

【産業上の利用分野】

本発明は高級言語プログラムを機械語プログラムに変換するプログラム変換装置、及び、その機械語プログラムを実行するプロセッサに関する。

【0002】

【従来の技術】

近年の電子技術の発展により、マイクロプロセッサ等の情報処理装置が普及し、あらゆる分野で用いられている。。

従来のプロセッサとしては、例えば、森下巖著、「マイクロプロセッサのハードウェア」（1984年11月9日、岩波書店）に示されている、セグメントアドレス方式の16ビットプロセッサがある。

【0003】

このセグメントアドレス方式の16ビットプロセッサは、16ビットよりも上位のビットを含む上位アドレスを格納するセグメントレジスタを備えることにより、データのビット幅が16ビットでありながら64キロバイトを超えるアドレス空間を扱うことができる。すなわち、64キロバイトを越える全アドレス空間をセグメントと呼ぶ64キロバイト単位の部分に分けるとともに、各セグメントに番号を付ける方式である。アドレスは、セグメントレジスタに収められたセグ

メント番号と、データのビット幅と同じ16ビットで表されるセグメントの先頭からの距離（オフセット）とで管理される。これにより、セグメントアドレス方式の16ビットプロセッサでは、データのビット幅が16ビットで64キロバイトを超えるアドレス空間を扱うことができる。

【0004】

また、従来のプロセッサとして、同書に示されている32ビットプロセッサがある。この32ビットプロセッサは、32ビット幅のアドレスを32ビット幅のデータと同一に管理して、4ギガバイトのアドレス空間を扱うことができる。

このようなプロセッサ上で実行される機械語プログラムは、コンパイラ等のプログラム変換装置によって生成される。

【0005】

上記16ビットプロセッサに対するコンパイラは、アドレス管理の方式に関して、ラージモデル方式と呼ばれるものと、ニアファーマデル方式と呼ばれるものがある。

ラージモデル方式を採用するコンパイラは、ポインタ変数を常にセグメントレジスタと16ビットオフセットとを対にして設定する。このため、このコンパイラによるオブジェクトコードを実行するプロセッサは、アドレス計算の度にセグメントレジスタの内容を計算して更新する操作が発生し、セグメントアドレス方式を採らない16ビットプロセッサに比べて著しく性能が劣化するという短所を有している。

【0006】

また、ニアファーマデル方式とを採用するコンパイラは、ラージモデル方式における上記の短所を次のようにして解決している。この方式は、同一セグメント内をアクセスするためのニアポインタ変数と、セグメント境界をまたいでアクセスするためのファーマポインタ変数という2種類のポインタ変数を指定できるようになっている。コンパイラは、ニアポインタ変数には16ビットオフセットのみを設定し、ファーマポインタ変数にはセグメントレジスタと16ビットオフセットとを対にして設定する。ニアポインタ変数とファーマポインタ変数のいずれを用いるかは、プログラマが選択する。

【0007】

このニアファーマデル方式は、ニアポインタ変数を用いる場面ではラージモデル方式に比べて性能が向上する反面、プログラマがプログラム作成時にセグメント境界を認識して2種類のポインタ変数を使い分ける必要があり、プログラム開発効率が著しく低下するという新たな問題点を有している。

他方、32ビットプロセッサに対するコンパイラは、上記の問題点を有していない。即ち、コンパイラが32ビットのデータ変数と同様にポインタ変数にも32ビットのアドレスを設定するので、プログラマがセグメントの境界を認識する必要がないので、プログラムの開発効率を低下させることもない。また、コンパイルされたプログラムを実行するプロセッサは、アドレス計算による性能劣化を発生させない。そして、4ギガバイトという広大なアドレス空間を扱うことができる。

【0008】

【発明が解決しようとする課題】

しかしながら従来技術のプロセッサおよびプログラム変換装置によれば、組み込み用途のマイクロコンピュータでの応用（アプリケーション）においては、32ビットのデータ幅を必要とせず、かつ、4ギガバイトのアドレス空間を必要としないものがほとんどである。多くのアプリケーションでは、データ幅については16ビットあれば十分であるが、アドレス空間については64キロバイトでは不足する。このような現状に鑑みると、データのビット幅が16ビット必要で、かつ、アドレス空間が64キロバイト以上を必要とするが4ギガバイトもの大きさを必要としない応用では、最適なマイクロコンピュータおよびコンパイラがなかったという問題点があった。

【0009】

より具体的に説明すると、32ビットプロセッサは、32ビットのデータおよびアドレスを処理するためのハードウェアを有することから、過大なハードウェアとなるのでコストおよび消費電力が増大するという問題点を有している。16ビットプロセッサは、アドレス空間の管理が従来技術で述べたようにアドレス計算の点で性能劣化を避けられない。また、プログラマがセグメントの境界を認識

する必要があることから、プログラムの開発効率が落ちる。

【0010】

また、32ビットプロセッサ用コンパイラは、16ビットまたは32ビットを基本語長とする機械語命令を使い分け、プログラム中では32ビットのアドレス指定が必要なために、プログラムのコードサイズが増大する。また、32ビットプロセッサとメモリとを接続するデータバスのビット幅が16ビットの場合は32ビットに比べて性能が著しく劣化するという問題点も有している。

【0011】

本発明は上記問題点に鑑み、データのビット幅が16ビットで16メガバイト程度のアドレス空間を必要とする応用において、好適したプログラム変換装置およびプロセッサを提供することを目的とする。

【0012】

【課題を解決するための手段】

上記の課題を解決するため本発明のプログラム変換装置は、

主たるデータの型における有効なビット幅よりも長いビット幅をもつアドレスを扱うプロセッサを対象とし、高級言語プログラムに基づいて機械語命令を生成するプログラム変換装置であって、

主たるデータの型における有効なビット幅およびアドレスのビット幅を保持するパラメータ保持手段と、

生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成し、アドレスを表す変数である場合、パラメータ保持手段が保持するアドレスのビット幅を有効とする命令を生成する生成手段とを備えている。

【0013】

前記データのビット幅Mは16であり、前記アドレスのビット幅Nは17以上31以下であってもよい。

また、前記生成手段は、

生成すべき機械語命令がメモリをアクセスする命令である場合に、アクセス対

象となる変数がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅をアクセス幅とすべきことを指示するメモリ管理手段と、

生成すべき機械語命令がレジスタを使用する命令である場合に、そのレジスタにリード又はライトされる変数がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅を有効とすべきことを指示するレジスタ管理手段と、

生成すべき機械語命令が即値を使用する命令である場合にその即値がデータであるかアドレスであるか応じて、パラメータ保持手段に保持された対応するビット幅の即値を用いるべきことを指示する即値管理手段と

メモリ管理手段、レジスタ管理手段、即値管理手段からの指示に従って、機械後命令を生成するコード生成手段と
からなっているもよい。

【0014】

また、本発明のプロセッサは、

上記プログラム変換装置により生成されたプログラムを実行するプロセッサであって、

Nビット長のプログラムカウンタと、

Nビット長の演算、及びデータのビット幅であるMビット長の演算を実行する実行手段と
を備えている。

【0015】

ここで、前記Nは24であり、Mは16であってもよい。

【0016】

【作用】

上記の手段により本発明のプログラム変換装置において、生成手段は、生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成する。アドレスを表す変数である場合、パラメータ保持手段が保持するアドレスのビット幅を有効

とする命令を生成する。

【0017】

また、前記生成手段において、メモリ管理手段は、生成すべき機械語命令がメモリをアクセスする命令である場合に、アクセス対象となる変数がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅をアクセス幅とすべきことをコード生成手段に指示する。レジスタ管理手段は、生成すべき機械語命令がレジスタを使用する命令である場合に、そのレジスタにリード又はライトされる変数がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅を有効とすべきことをコード生成手段に指示する。即値管理手段は、生成すべき機械語命令が即値を使用する命令である場合にその即値がデータであるかアドレスであるかに応じて、パラメータ保持手段に保持された対応するビット幅の即値を用いるべきことをコード生成手段に指示する。

【0018】

また、本発明のプロセッサは、N(24)ビット長のプログラムカウンタにより2のN乗(16メガバイト)のアドレス空間を利用でき、実行手段によりN(24)ビット長の演算、及びデータのビット幅であるM(16)ビット長の演算を実行し得る。

【0019】

【実施例】

図1は本発明の実施例におけるマイクロコンピュータおよびプログラム変換装置(以下、コンパイラと略す。マイクロコンピュータとコンパイラを含めて情報処理装置と呼ぶ)のブロック図を示す。図1において、本情報処理装置は、C言語プログラムを格納している記憶装置1と、C言語プログラムを機械語プログラムに翻訳するコンパイラ2と、オブジェクトコードを実行するハードウェア3とからなる。

【0020】

コンパイラ2は、以下のものからなる。

5は構文解析部で、C言語プログラムの構文を解析し、プログラム文が中間形

式に変換された中間形式文及び中間コードからなる中間ファイルを生成する。

7はコード生成部で、中間ファイルに基づいてオブジェクトコードを生成する。

【0021】

8はパラメータ設定部で、整数型のデータ変数のビット幅とポインタ変数のビット幅とを保持する。本実施例では、予め利用者により整数型のデータ変数のビット幅が16ビットに、ポインタ変数のビット幅が24ビットに設定されているものとする。

9はメモリ管理部で、コード生成部7が生成すべきロード・ストア命令について、その命令の対象となる変数の型に応じて、メモリを読み書きするビット幅を指示する。

【0022】

10はレジスタ管理部で、コード生成部7が生成すべきレジスタを使用する命令について、レジスタにリード／ライトされる変数の型に応じて、有効なビット幅を指示する。

11は即値管理部で、コード生成部7が生成すべき即値を用いる命令について、即値である変数の型に応じて、命令コード中の即値のビット幅を指示する。

【0023】

また、ハードウェア3は、以下のものからなる。

13はメモリで、オブジェクトコードおよびプログラムで用いられるデータを格納する。

14はアドレスバスで、24ビットのビット幅を持つ。

15はデータバスで、16ビットのビット幅を持つ。

【0024】

16はバス制御回路で、アドレスバス14とデータバス15とを介してメモリ13に接続され、メモリ13に格納されたオブジェクトコードを命令として逐一読み出し、またメモリ13に対してデータを読み書きする。

17は命令制御回路で、プログラムカウンタ19を有し、バス制御回路16へ命令アドレスを指示し、バス制御回路16から命令を受け取り解読する。

【0025】

プログラムカウンタ19は、24ビットの命令アドレスを出力する。

18は演算実行回路で、レジスタファイル20と演算器21からなる。

レジスタファイル20は、24ビットの複数のレジスタを有する。

演算器21は、レジスタファイル20に保持された値を用いて24ビット幅の算術論理演算等を実行する。

【0026】

図2は、コード生成部7の処理フローを示す。

ステップ20は、中間ファイルからデータ変数を抽出して、それぞれの変数に関する情報を載せた変数テーブル（シンボルテーブル）を作成する。シンボルテーブルの一例を図7に示す。同図において、シンボル欄はデータ変数のシンボルを、型欄はデータ変数が整数型であるかポインタ型であるかを、バイト数欄はそのデータ変数のバイト数を、先頭アドレス欄はメモリに割り付けられている場合の先頭アドレスを、レジスタ欄はレジスタが割り当てられている場合のレジスタ名を示す。

【0027】

ステップ21は、中間ファイルからコード生成が未処理の中間命令があるか否かを判定する。あればステップ22に進み、なければコード生成が終了する。

ステップ22は、中間ファイルから次に処理すべき順の中間命令を1つ読み出す。

ステップ23は、1つの中間命令を実現する1つ又は複数の機械語命令を選定する。

【0028】

ステップ24は、選定された機械語命令から次に処理すべき順の機械語命令を1つ指定する。

ステップ25は、指定された1つの機械語命令（以下、個別命令と呼ぶ）の命令コードを生成する処理（以下、個別処理と呼ぶ）を行う。

ステップ26は、ステップ23で選定された機械語命令のうち次に処理すべき機械語命令があるか否かを判定する。あればステップ24に戻り、なければステ

ップ21に戻る。

【0029】

図3～図6は、上記ステップ25の個別処理のフローを示す。

第1に、ステップ30は、個別命令がロード／ストア命令の種類に属するか否かを判定し、そうである場合はメモリ管理部9に通知し、そうでない場合ステップ31に進む。この通知を受けたメモリ管理部9の処理フローを図4に示す。

図4において、ステップ40は、その個別命令（ロード／ストア命令）においてメモリとの間でロード／ストアすべき変数について、シンボルテーブルを参照し、その変数の型を調べる。

【0030】

ステップ41は、その変数の型が整数型データ変数であるか、ポインタ型データ変数であるかを判定し、前者である場合ステップ42に、後者である場合ステップ43に進む。

ステップ42は、アクセスするデータ幅が2バイトのロード／ストア命令を生成すべきことをコード生成部7に指示し、ステップ31に進む。

【0031】

ステップ43は、アクセスするデータ幅が3バイトのロード／ストア命令を生成すべきことをコード生成部7に指示し、ステップ31に進む。

第2に、ステップ31は、個別命令がレジスタを使用するか否かを判定し、そうである場合はレジスタ管理部10に通知し、そうでない場合ステップ32に進む。この通知を受けた場合のレジスタ管理部10の処理フローを図5に示す。

図5において、ステップ50は、その個別命令が使用するレジスタに格納される変数について、シンボルテーブルを参照し、その変数の型を調べる。

【0032】

ステップ51は、その変数の型が整数型データ変数であるか、ポインタ型データ変数であるかを判定し、前者である場合ステップ52に、後者である場合ステップ53に進む。

ステップ52は、使用するレジスタの下位16ビットを有効とする命令を生成すべきことをコード生成部7に指示し、ステップ32に進む。

【0033】

ステップ53は、使用するレジスタの全24ビットを有効とする命令を生成すべきことをコード生成部7に指示し、ステップ32に進む。

第3に、ステップ32は、個別命令が即値データを使用するか否かを判定し、そうである場合は即値管理部11に通知し、そうでない場合ステップ33に進む。この通知を受けた場合の即値管理部11の処理フローを図6に示す。

【0034】

図6において、ステップ60は、当該個別命令において使用する即値データとなる変数について、シンボルテーブルを参照し、その変数の型を調べる。

ステップ61は、その変数の型が整数型データ変数であるか、ポインタ型データ変数であるかを判定し、前者である場合ステップ62に、後者である場合ステップ63に進む。

【0035】

ステップ62は、2バイトの即値を持つ命令を生成すべきことをコード生成部7に指示し、ステップ33に進む。

ステップ63は、3バイトの即値を持つ命令を生成すべきことをコード生成部7に指示し、ステップ33に進む。

第4に、ステップ33は、メモリ管理部9、レジスタ管理部10、即値管理部11からの指示があれば、それにしたがって個別命令の命令コードを生成する。

【0036】

以上のように構成された本実施例のマイクロコンピュータおよびプログラム変換装置について、以下その動作を説明する。

説明を簡潔にするために、記憶装置1に記憶されているC言語プログラムが次に示す場合を例にあげる。

```
main ( )
{
    int *a, b, c ;
    c = *a + b + 1 ;
}
```


構文解析部 5 は、記憶装置 1 から C 言語プログラムを取り出し、構文を文法に照らして解析して中間形式の言語で記述された中間ファイルを生成する。この中間ファイルのイメージを以下に示す。ただし、ここでは便宜上、中間形式の記述を意味がわかるように書き直してある。

【0037】

中間形式文 1 : (int *a, b, c)

中間命令 1 : t1:=*a

中間命令 2 : t2:=t1+b

中間命令 3 : t3:=t2+1

中間命令 4 : c:=t3

中間形式文 1 は、宣言文 `int *a, b, c` に対応し、中間命令 1 ~ 4 は、演算式 `c = *a + b + 1` に対応する。

【0038】

これらの中間形式文、中間命令は、それぞれ以下のようにしてオブジェクトコードに変換される。

この中間ファイルが入力されると、コード生成部 7 は、中間ファイル中のデータ変数（宣言のないものも含めて）を抽出し、それぞれの変数の型を調査し、必要があればメモリに割り当て、図 7 に示したシンボルテーブルを作成する（図 2 のステップ 20）。

【0039】

上の例では、中間形式文 1 から明示に宣言された変数 `*a, b, c` が抽出される。ポインタ変数として宣言された変数 `*a` は、パラメータ設定部 8 においてポインタ変数のビット幅が 24 ビットに設定されているので、メモリ上に 24 ビット（3 バイト）の領域が確保され割り付けられる。また、整数型データ変数として宣言された 2 つの変数 `b, c` は、パラメータ設定部 8 において整数型データ変数のビット幅が 16 ビットに設定されているので、それぞれメモリ上に 16 ビット（2 バイト）の領域が確保され割り付けられる。ここでは、変数 `*a, b, c` は、それぞれメモリの 1000 番地から 3 バイト、1004 番地から 2 バイト、1006 番地から 2 バイトの領域に割り当てられているものとする。ただし、1003 番地の 1 バイトは空領域であ

る。

【0040】

さらに、中間命令1～4から一時変数t1,t2,t3が抽出される。一時変数t1,t2,t3は、演算対象となる変数に合わせて、それぞれ整数型データ変数として扱われる。

これらの変数に関する情報がシンボルテーブルに書き込まれる。このシンボルテーブルの内容は、以後レジスタ割当等の変更があれば、その都度ダイナミックにその内容が書き換えられる。既を示した図7は、この時点におけるシンボルテーブルの内容である。この時点では、レジスタ欄及び一時変数の先頭アドレス欄は、割当がないので空欄のままである。

【0041】

この後、コード生成部7は、中間命令のそれぞれについて、以下のようにして対応する機械語命令を生成する。

<< 中間命令1 : t1:=*a >>

コード生成部7は、中間ファイル中に処理すべき（未処理の）中間命令が残っているか否かを判定する（図2のステップ21）。この時点では、中間命令1～4が残っているので、ステップ22に進む。

【0042】

次に、コード生成部7は、未処理の中間命令のうち先頭の中間命令を1つ読み出し（ステップ22）、その中間命令を実現する1つ又は複数の機械語命令を選定する（ステップ23）。

ステップ23における中間命令1を実現する1つ又は複数の命令の選定を具体的に説明する。

【0043】

中間命令1は「①ポインタ変数*aが割り当てられている1000番地から、②3バイトを読み出して、③その内容をアドレスとして2バイト読み出して一時変数t1に格納する。」ことを内容とする。コード生成部7は、この内容を実現するため上記①～③に対応する機械語命令を選定する。すなわち、①アドレス1000番地を即値データとして、第1のアドレスレジスタに格納するmov命令、②第1のアド

レジスタを用いて*aの内容（ポインタ）をアドレスとして、第2のアドレスレジスタに読み出すmov命令、③第2のアドレスレジスタを用いて、ポインタが指すデータをデータレジスタに読み出すmov命令、の3つの命令を選定する。上記第1、第2のアドレスレジスタは、A0、A1が割り当てられているものとする。これは、シンボルテーブルに記入される。

【0044】

＜個別命令①の生成＞

さらに、コード生成部7は、これらの複数の命令の中から処理すべき命令（個別命令）を1つ指定し（ステップ24）、個別命令①に対する命令コードを生成する（ステップ25）。

このステップ25における個別命令①の命令コード生成を図3～6のフローを用いて具体的に説明する。

【0045】

コード生成部7は、ステップ24で指定された個別命令①がメモリアクセスするロード／ストア命令でなく（図3のステップ30）、レジスタA0を使用する命令であるのでレジスタ管理部10にその旨通知する（ステップ31）。

レジスタ管理部10は、シンボルテーブルを参照し（図5のステップ50）、レジスタA0に格納すべき変数がポインタであることから（ステップ51）、レジスタの全24ビットが有効となる命令を生成すべきことをコード生成部7に対して指示する（ステップ53）。

【0046】

さらに、コード生成部7は、個別命令①が即値データを用いるので即値管理部11にその旨通知する（ステップ32）。即値管理部11は、シンボルテーブルを参照し（図6のステップ60）、レジスタA0に格納すべき即値がポインタであることから（ステップ61）、3バイトの即値データを有する命令を生成すべきことをコード生成部7に対して指示する（ステップ63）。

【0047】

ステップ53、63からの指示に従って、コード生成部7は、個別命令①に対応する、次に示す命令1を生成する（ステップ33）。

命令1: MOV #001000,A0

この後コード生成部7は、ステップ23で選定された複数命令のうち、次の命令コードを生成すべき命令②、③が残っているので（ステップ26）、ステップ24に進む。

【0048】

＜個別命令②の生成＞

コード生成部7は、これらの複数の命令の中から処理すべき命令（個別命令）を1つ指定し（ステップ24）、個別命令②に対する命令コードを生成する（ステップ25）。

このステップ25における個別命令①の命令コード生成を図3～6のフローを用いて具体的に説明する。

【0049】

コード生成部7は、個別命令②がメモリアクセスするロード／ストア命令であるので、メモリ管理部9にその旨通知する（図3のステップ30）。

メモリ管理部9は、シンボルテーブルを参照し（図4のステップ40）、第2のレジスタA1に格納すべき変数がポインタであることから（ステップ41）、メモリアクセスにおけるアクセス幅が3バイトであるロード命令を生成すべきことをコード生成部7に対して指示する（ステップ43）。

【0050】

さらに、コード生成部7は、個別命令②がレジスタA0、A1を使用する命令であるのでレジスタ管理部10にその旨通知する（ステップ31）。

レジスタ管理部10は、シンボルテーブルを参照し（図5のステップ50）、第2のレジスタA1に格納すべき変数がポインタであることから（ステップ51）、レジスタの全24ビットが有効となる命令を生成すべきことをコード生成部7に対して指示する（ステップ53）。

【0051】

この後、コード生成部7は、個別命令②が即値データを用いないので（ステップ32）、ステップ33に進む。

ステップ43、53からの指示に従って、コード生成部7は個別命令②に対応

する、次に示す命令2を生成する（ステップ33）。

命令2： MOV @A0,A1

この後コード生成部7は、ステップ23で選定された複数命令のうち、次の命令コードを生成すべき命令③が残っているので（ステップ26）、ステップ24に進む。

【0052】

<個別命令③の生成>

個別命令③についても上記と同様にして、コード生成部7は次の命令3を生成する（ステップ24～26）。この時点で、一時変数t1には、D0レジスタが割り当てられている。

命令3： MOV @A1,D0

以下、上記と同様にして、ステップ21～26のループにおいて各中間命令の処理が、ステップ24、25（図3、4のフロー）、26において各個別命令の処理が行われるので、詳細な説明は省略して概要を示す。

【0053】

<< 中間命令2：t2:=t1+b >>

この中間命令は、「④変数bが割り当てられている1004番地から2バイト読み出して、⑤その内容と一時変数t1とを加算して一時変数t2に格納する。」ことを内容とする。

まず、コード生成部7は、個別命令④に対応してA0レジスタの内容が示す1000番地から4番地離れた番地をロードする命令4を生成する。その際、メモリ管理部9は、シンボルテーブルを参照して（ステップ40）、個別命令4が整数型データ変数をロードすることから（ステップ41）、アクセス幅が2バイトのロード命令を生成すべきことを指示する（ステップ42）。レジスタ管理部10は、シンボルテーブルを参照して（ステップ50）、レジスタに整数型データを格納するので（ステップ51）、レジスタの下位16ビットが有効となる命令を生成すべきことを指示する（ステップ52）。この時点で、レジスタD1には変数bが格納されている。

【0054】

命令4 : MOV @(04,A0),D1

次に、コード生成部7は、個別命令⑤に対応してD1レジスタに格納された変数bと、D0レジスタの内容が示す一時変数t1とを加算して、その結果をD1レジスタに格納する命令5を生成する。その際、レジスタ管理部10は、シンボルテーブルを参照して（ステップ50）、レジスタに整数型データを格納するので（ステップ51）、レジスタの下位16ビットが有効となる命令を生成すべきことを指示する（ステップ52）。この時点で、一時変数t2にはレジスタD1が割り当てられている。

【0055】

命令5 : ADD D0,D1

<<中間命令3 : t3:=t2+1 >>

この中間命令は、「⑥一時変数t2に1を加算して一時変数t3に格納する」ことを内容とする。コード生成部7は、レジスタ管理部10、即値管理部11の指示に従って、2バイトの即値#0001とレジスタD1とを加算した結果をD1レジスタに格納する次の命令6を生成する。この時点で、一時変数t3にはレジスタD1が割り当てられている。

【0056】

命令6 : ADD #0001,D1

<<中間命令4 : c:=t3 >>

この中間命令は「⑦変数cが割り当てられている1006番地から2バイトの領域に一時変数t3を書き込む」ことを内容とする。コード生成部7は、メモリ管理部9の指示に従って、レジスタA1の内容が示す番地から6番地離れた番地にレジスタD1の内容をストアする次の命令7を生成する。

【0057】

命令7 : MOV D1,@(06,A0)

上記のようにして各中間命令の処理が終了する。その結果、コード生成部7から次のようなオブジェクトコードがメモリ13に対して出力される。メモリ13上で、命令1から命令7はそれぞれ、100000番地、100005番地、100007番地、100008番地、10000a番地、10000b番地、10000f番地に配置されるものとする。

【0058】

命令1:	100000番地	MOV	#001000,A0
命令2:	100005番地	MOV	@A0,A1
命令3:	100007番地	MOV	@A1,D0
命令4:	100008番地	MOV	@(04,A0),D1
命令5:	10000a番地	ADD	D0,D1
命令6:	10000b番地	ADD	#0001,D1
命令7:	10000f番地	MOV	D1,@(06,A0)

これらの命令は、オブジェクトコードを便宜上ニモニク（アセンブリ言語）で表現したものであり、メモリ13に格納される際は2進数で表される。上記の数値はすべて16進数である。

以下、メモリ13に配置されたこのオブジェクトコードについて、マイクロコンピュータにより実行される動作を説明する。

【0059】

（命令1：100000番地 MOV #001000,A0 ）

バス制御回路16および命令制御回路17は、プログラムカウンタ19が保持している値100000をアドレスバス14に出力し、データバス15を介して命令1をフェッチして解読する。解読結果に従って演算実行回路18は、命令1のオペランドで指定されている即値001000を命令制御回路17から受け取り、レジスタファイル20の中のA0レジスタに格納する。

【0060】

（命令2：100005番地 MOV @A0,A1 ）

バス制御回路16および命令制御回路17は、同様にして、命令2をフェッチして解読する。演算実行回路18は、A0レジスタの内容を読み出し、それをアドレスバス14に出力してメモリ13を読み出し、データバス15を介して読み出された16ビットのデータをA1レジスタの下位16ビットに格納する。続いて、演算実行回路18は、演算器21によりA0レジスタの値001000に2を加算し、バス制御回路16により加算結果001002をアドレスバス14に出力してメモリ13を読み出し、データバス15を介して読み出された8ビットのデータをA1レジス

タの上位8ビットに格納する。

【0061】

(命令3:100007番地 MOV @A1,D0)

バス制御回路16および命令制御回路17は、命令3をフェッチして解読する。演算実行回路18は、A1レジスタの内容を読み出し、それをアドレスバス14に出力し、メモリ13を読み出す。その後演算実行回路18は、データバス15を介して読み出された16ビットの値をD0レジスタの下位16ビットに格納する。D0レジスタは、ポインタ変数*aが指し示すデータを保持することになる。

【0062】

(命令4:100008番地 MOV @(04,A0),D1)

バス制御回路16および命令制御回路17は、命令4をフェッチして解読する。演算実行回路18は、変位値04を命令制御回路17から受け取り、演算器21において読み出されたA0レジスタの値001000と加算し、加算結果001004をバス制御回路16によりアドレスバス14に出力しメモリ13の読み出しを行う。その後演算実行回路18は、データバス15に読み出された16ビットの値をレジスタファイル20の中のD1レジスタの下位16ビットに格納する。D1レジスタは、変数bを保持することになる。

【0063】

(命令5:10000a番地 ADD D0,D1)

バス制御回路16および命令制御回路17は、命令5をフェッチして解読する。演算実行回路18は、演算器21においてレジスタファイル20から読み出されたD0レジスタの値とD1レジスタの値とを加算し、加算結果をレジスタファイル20の中のD1レジスタに格納する。演算器21はこの加算を24ビットで行うが、D1レジスタは下位16ビットが有効である。D1レジスタは、ポインタ変数*aが示すデータと変数bとの加算値を保持することになる。

【0064】

(命令6:10000b番地 ADD #0001,D1)

バス制御回路16および命令制御回路17は、命令6をフェッチして解読する。演算実行回路18は、演算器21においてレジスタファイル20から読み出さ

れたD1レジスタの値と、命令制御回路17から受け取った即値0001とを加算し、加算結果をD1レジスタに格納する。D1レジスタは、ポインタ変数*aが示すデータと変数bと即値0001との加算値を保持することになる。

【0065】

(命令7:10000f番地 MOV D1,@(06,A0))

バス制御回路16および命令制御回路17は、命令7をフェッチして解読する。演算実行回路18は、変位置06を命令制御回路17から受け取り、演算器21においてレジスタファイル20から読み出されたA0レジスタの値001000と加算し、加算結果001006をバス制御回路16によりデータバス15に出力するとともに、D1レジスタの下位16ビットの値をデータバス15に出力し、メモリ13に対して16ビットデータの書き込みを行う。メモリ13の001006番地には、ポインタ変数*aが示すデータと変数bと即値0001との加算値が書き込まれることになる。

【0066】

このようにしてC言語プログラムがコンパイラ2で翻訳されて生成されたオブジェクトコードが、ハードウェア3で実行される。

以上のように本実施例によれば、コンパイラ2が全ての変数と生成するオブジェクトコードとのメモリ上での配置を24ビットのアドレスで管理し、マイクロコンピュータが24ビットのレジスタファイル20および演算器21でこれらのアドレスを計算し24ビットのアドレスバス14でメモリ13をアクセスするため、本実施例の情報処理装置はセグメントに分割しない全く均一な16メガバイトのアドレス空間を実現することができ、従ってC言語プログラムを記述するプログラムはセグメントの境界のような空間の不均一性を認識する必要がなく、またセグメントレジスタの操作のようなアドレス計算での性能劣化も発生しない。それに伴ってプログラムの開発効率も向上する。

【0067】

また、データのビット幅が16ビットで16メガバイトのアドレス空間を必要とする応用にとって、本実施例の情報処理装置は、マイクロコンピュータが24ビットのレジスタファイル20および演算器21で構成されるため、従来の32

ビットプロセッサのように過大なハードウェアによってコストおよび消費電力が増大することがない。

【0068】

また、マイクロコンピュータの機械語命令の基本語長は8ビットであり、上記の命令1のように命令中にアドレスの即値が伴う場合でもその即値は最大でも24ビットとなるため、コンパイラ2が生成するオブジェクトコードは、機械語命令の基本語長を16または32ビットとしアドレスの即値が最大32ビットとなる32ビットプロセッサの情報処理装置に比べて極めて小さくなる。また64キロバイトのアドレス空間しか扱えない16ビットプロセッサに比べても、コードサイズの増加の要因が命令に伴うアドレスの即値による最大1バイトだけであるため、オブジェクトコードのサイズはほとんど大きくなる。

【0069】

本実施例のマイクロコンピュータのデータバス15の幅は16ビットであるが、これを24ビットとした場合と比べると、レジスタファイル20に格納された24ビットのアドレスに関する値をメモリ13に対して読み書きする時のみ実行時間が長くなるだけであり、この性能の低下は、メモリに対して常に32ビットで読み書きする32ビットプロセッサの情報処理装置のデータバスのビット幅を16ビットにした場合に比べて極めて小さい。

【0070】

なお、本実施例は、マイクロコンピュータのアドレスバス14、プログラムカウンタ19、レジスタファイル20および演算器21のビット幅を24ビットとし、コンパイラ2のパラメータ設定部8のポインタ変数のビット幅を24ビットに設定しているが、これらを必要とするアドレス空間の広さに対応して17ビットから31ビットの任意のビット幅にしてもよい。対応は次のようになる。

【0071】

必要なアドレス空間	ビット幅
128キロバイト	17ビット
256キロバイト	18ビット
512キロバイト	19ビット

1メガバイト	20ビット
2メガバイト	21ビット
4メガバイト	22ビット
8メガバイト	23ビット
16メガバイト	24ビット (本実施例)
32メガバイト	25ビット
64メガバイト	26ビット
128メガバイト	27ビット
256メガバイト	28ビット
512メガバイト	29ビット
1ギガバイト	30ビット
2ギガバイト	31ビット

このようにすることにより、アドレスのビット幅を越す過大なハードウェアがなくなり、応用に応じてコストおよび消費電力の最適化を図ることができる。

【0072】

また本実施例は、マイクロコンピュータのデータバス15の幅を16ビットとしているが、これを24ビットとしてもよい。

また本実施例は、マイクロコンピュータのアドレスバス14、プログラムカウンタ19、レジスタファイル20および演算器21のビット幅とコンパイラ2のパラメータ設定部8のポインタ変数のビット幅の設定値とを24ビットとし、コンパイラ2のパラメータ設定部8のデータ変数のビット幅の設定値を16ビットとしているが、これらのビット幅を限定するものではない。後者をプログラムが主として扱う型のデータのビット幅(Mとする)と等しくし、前者をMより大きい値(Nとする)とすることにより、情報処理装置は、主として扱う型のデータのビット幅で表現できる2のM乗バイトの空間を越える2のN乗バイトのアドレス空間を扱うことができる。

【0073】

また本実施例は、パラメータ設定部8でビット幅を設定するデータ変数の型を整数型としているが、これをC言語プログラムで主として扱うデータの型に合わ

せていかなる型にしてもよい。

また本実施例は、C言語プログラムをコンパイルして実行するものであるが、言語を限定するものではない。コンパイラ2をプログラムの記述言語に対応させることにより、情報処理装置はいかなる言語のプログラムでも実行できる。

【0074】

【発明の効果】

以上説明してきたように本発明によれば、パラメータ設定手段にデータ幅及びポインタ幅を任意に設定できるので、プログラマにセグメントの境界のような空間の不均一性を認識させることなく、かつセグメントレジスタの操作のようなアドレス計算での性能劣化をも伴うことなく、主として扱う型のデータのビット幅で表現できる空間を超える広さの任意のアドレス空間を扱うことができるという効果がある。

【0075】

また、組み込み用途のマイクロコンピュータでの応用（アプリケーション）においては、データのビット幅が16ビット必要で、かつ、アドレス空間が64キロバイト以上を必要とするが4ギガバイトもの大きさを必要としないものに対して最適化を図ることができるという効果がある。

また本発明によれば、従来の32ビットプロセッサの情報処理装置に比べてプログラムのコードサイズが小さく、またデータバスのビット幅が16ビットの場合でも性能劣化がほとんどない情報処理装置を実現できるという効果がある。

【図面の簡単な説明】

【図1】

本発明の実施例における情報処理装置のブロック図である。

【図2】

同実施例におけるコード生成部7の処理フローを示す。

【図3】

同実施例におけるコード生成部7の処理フローのうち個別処理のフローを示す。

【図4】

同実施例におけるメモリ管理部 9 の処理フローを示す。

【図 5】

同実施例におけるレジスタ管理部 10 の処理フローを示す。

【図 6】

同実施例における即値管理部 11 の処理フローを示す。

【図 7】

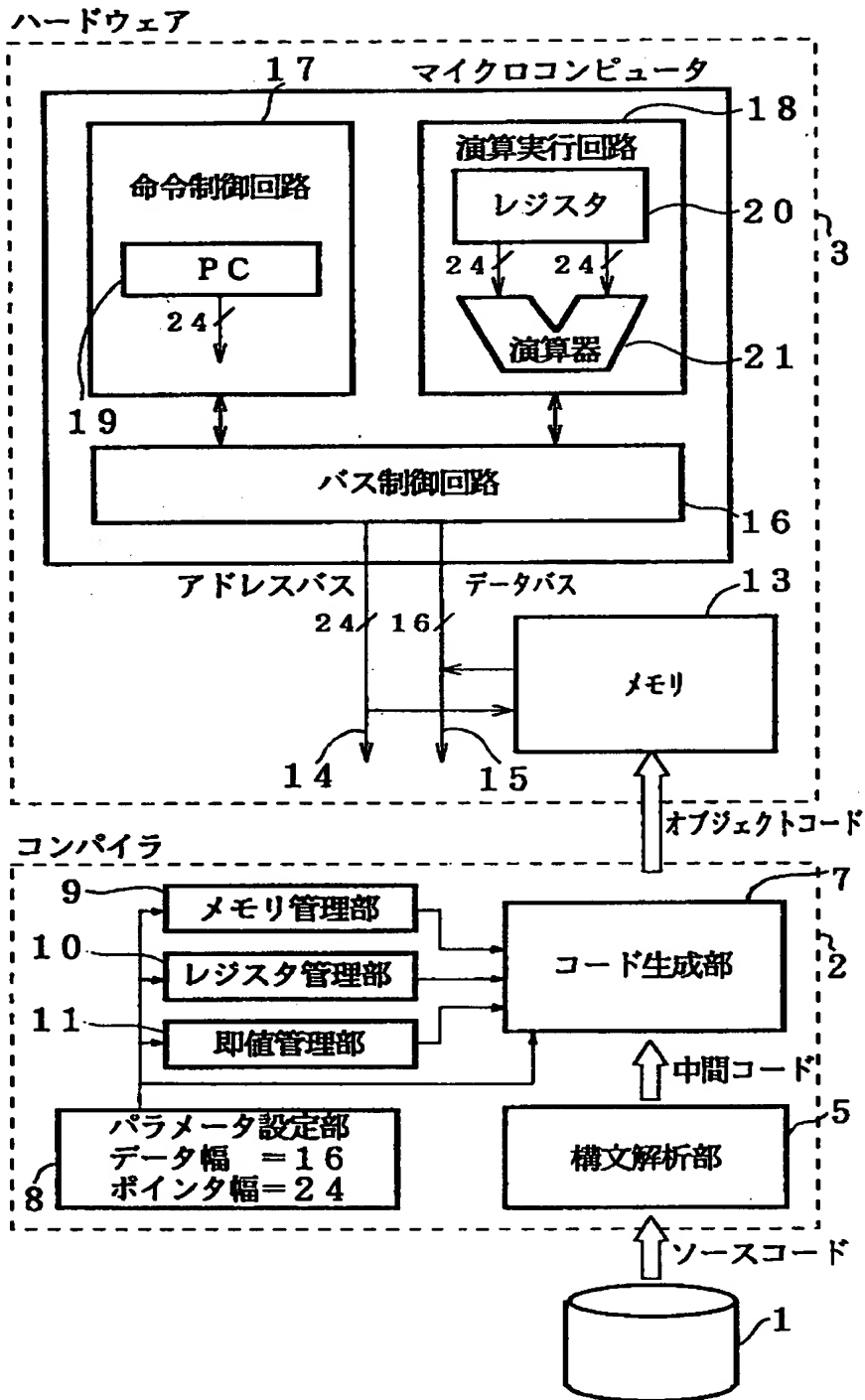
同実施例におけるシンボルテーブルを示す。

【符号の説明】

- 1 記憶装置
- 2 コンパイラ
- 3 ハードウェア
- 5 構文解析部
- 7 コード生成部
- 8 パラメータ設定部
- 9 メモリ管理部
- 10 レジスタ管理部
- 11 即値管理部
- 13 メモリ
- 14 アドレスバス
- 15 データバス
- 16 バス制御回路
- 17 命令制御回路
- 18 演算実行回路
- 19 プログラムカウンタ
- 20 レジスタファイル
- 21 演算器

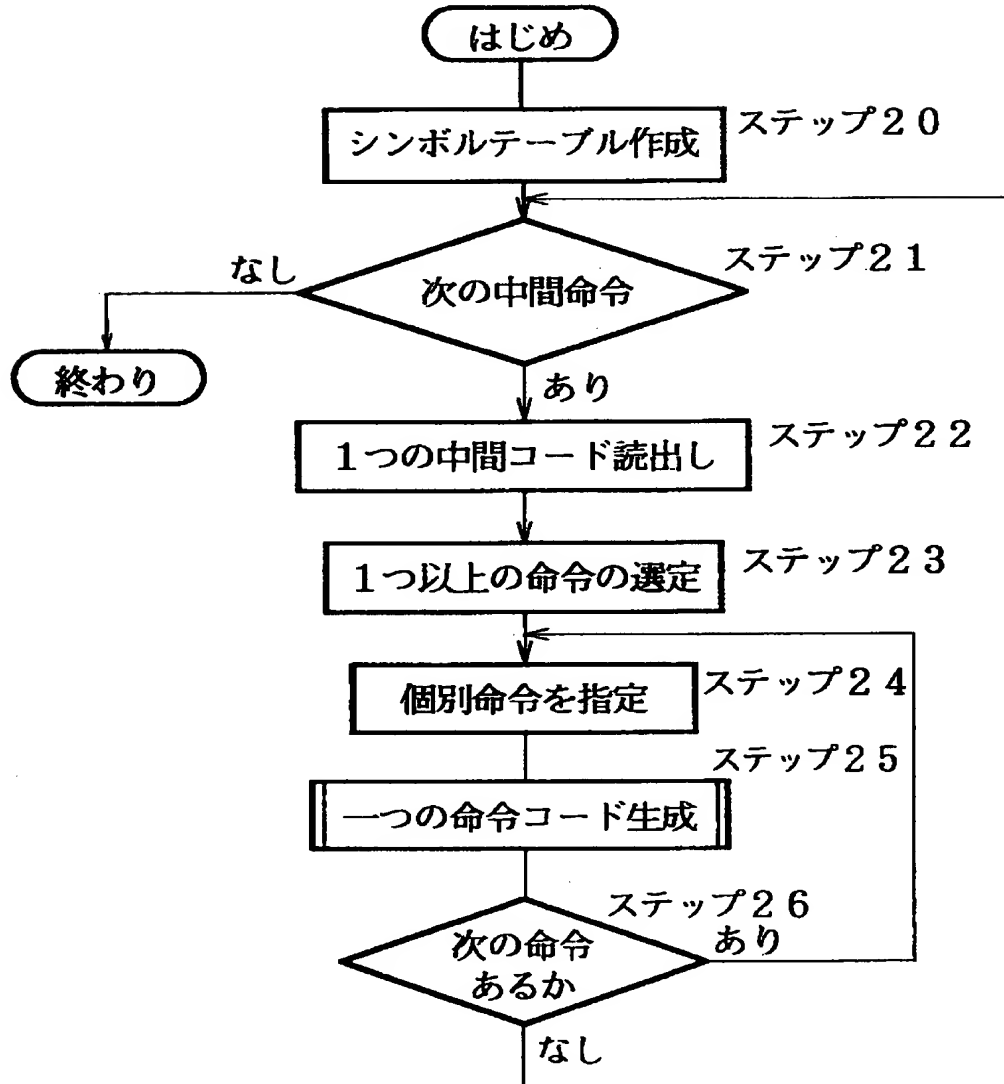
【書類名】 図面

【図1】



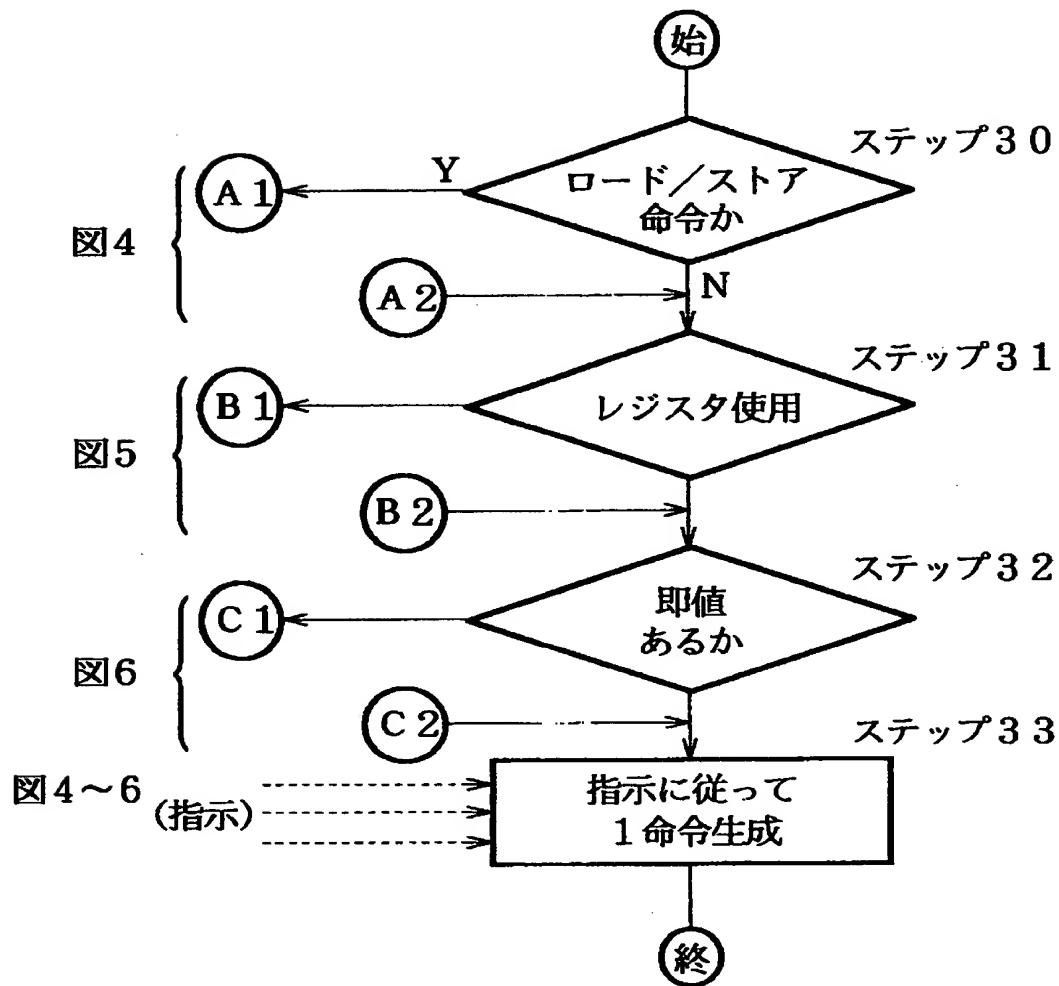
【図2】

コード生成部7の処理フロー

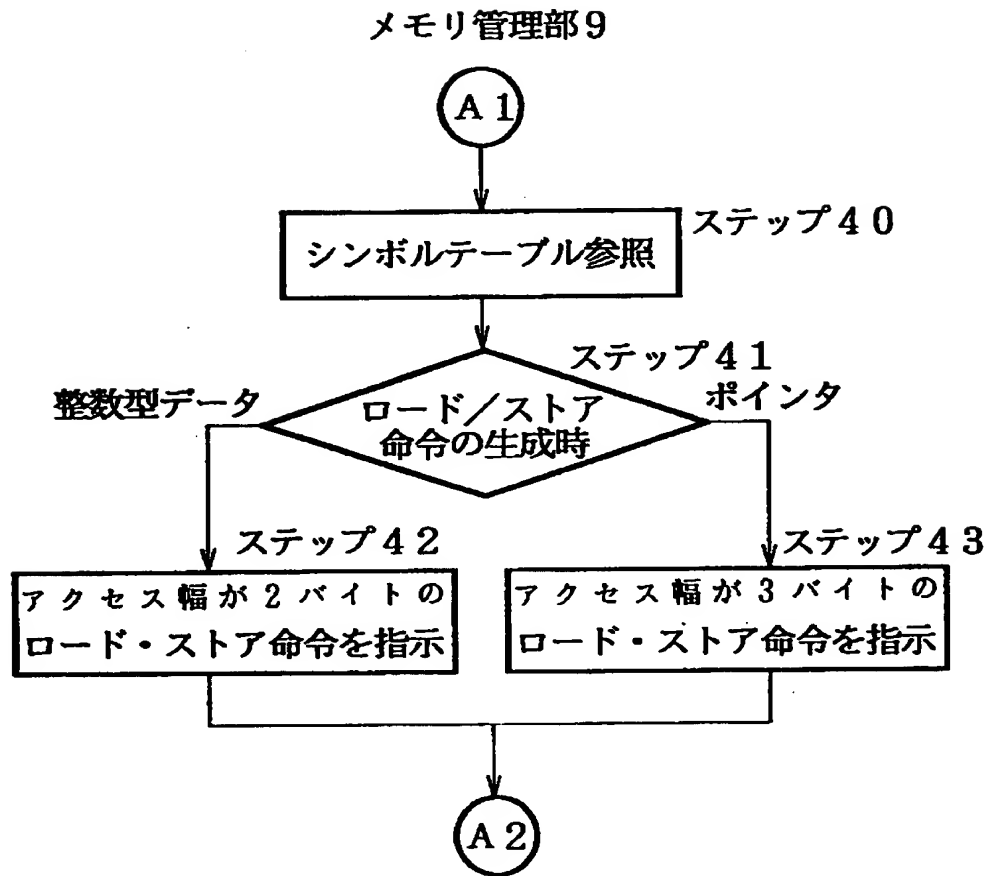


【図3】

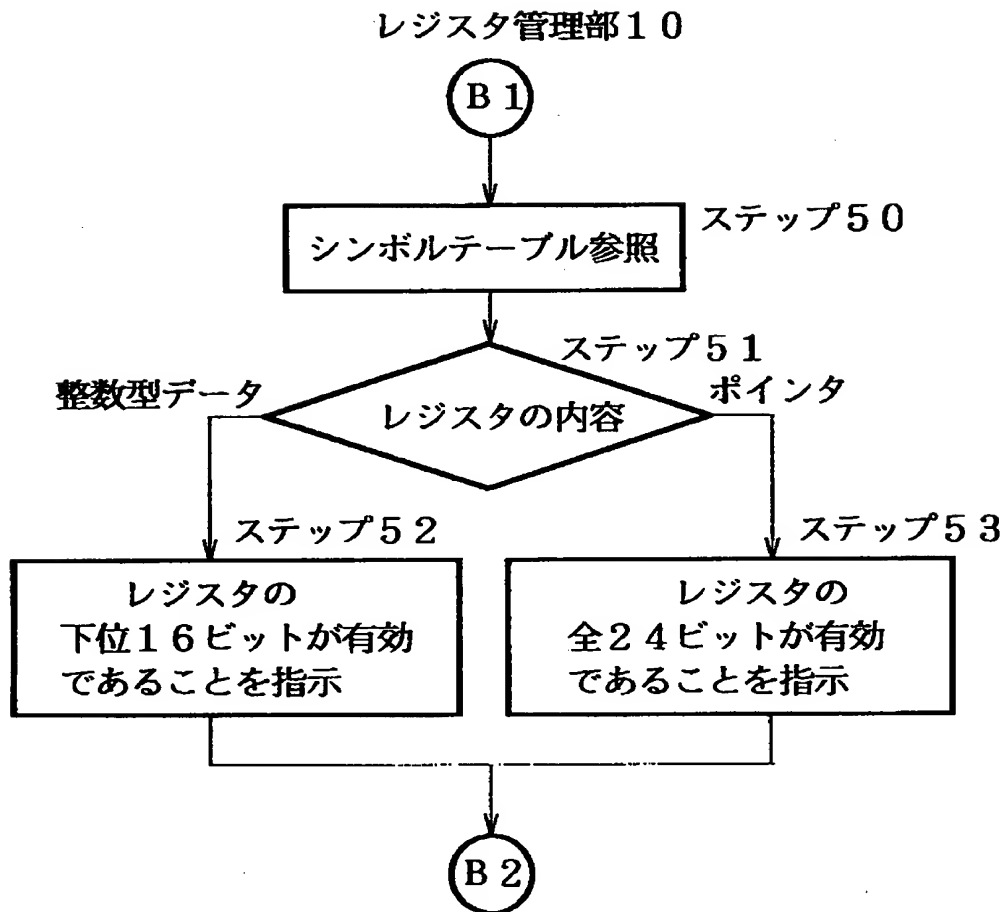
コード生成部のフロー



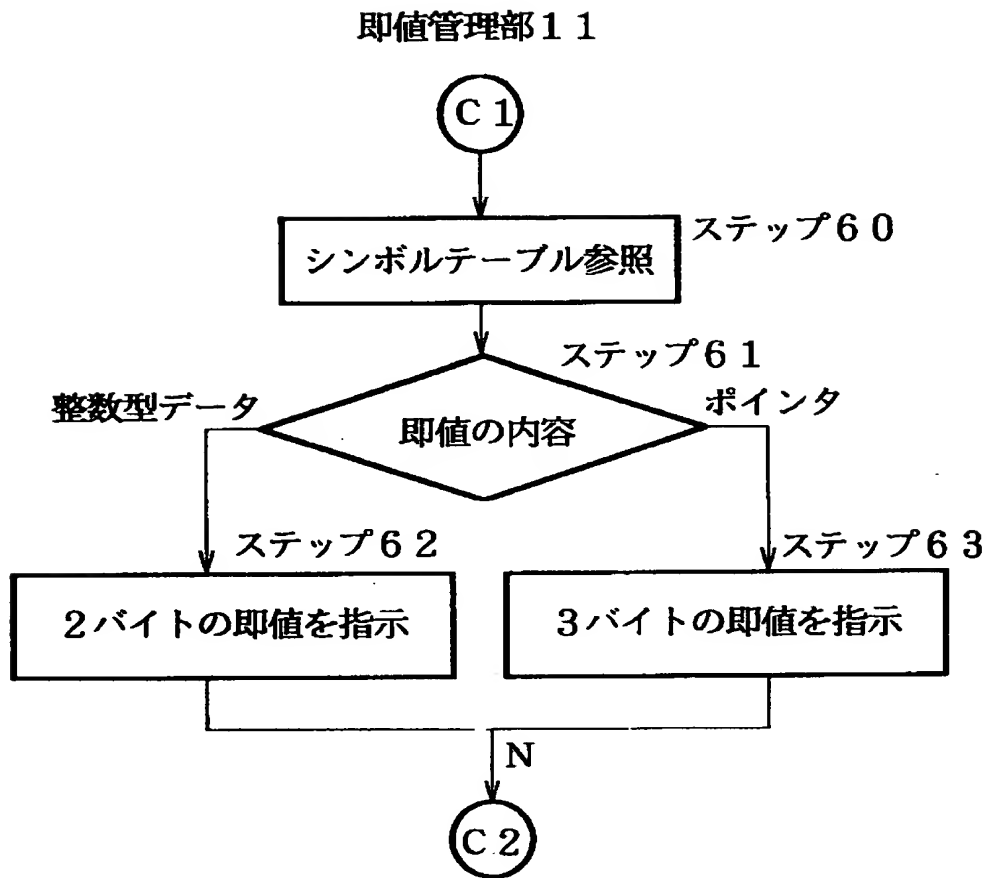
【図4】



【図5】



【図6】



【図 7】

シンボルテーブル

シンボル	型	バイト数	先頭アドレス	レジスタ
*a	ポインタ	3	1000	
b	整数	2	1004	
c	整数	2	1006	
t 1	整数	2		
t 2	整数	2		
t 3	整数	2		

【書類名】 要約書

【要約】

【目的】 本発明は、データのビット幅が16ビットで16メガバイト程度のアドレス空間を必要とする応用において、好適したコンパイラおよびマイクロコンピュータを提供することを目的とする。

【構成】 プログラム変換装置において、メモリ管理部9、レジスタ管理部10、即値管理部11からの指示により、コード生成部7は、生成すべき機械語命令で用いられる変数が、データを表す変数である場合、パラメータ保持手段が保持するデータのビット幅を有効とする命令を生成する。アドレスを表す変数である場合、パラメータ設定部8が保持するアドレスのビット幅を有効とする命令を生成する。マイクロコンピュータは、N(24)ビット長のプログラムカウンタにより2のN乗(16Mバイト)のアドレス空間を利用でき、実行手段によりN(24)ビット長の演算、及びデータのビット幅であるM(16)ビット長の演算を実行し得る。

【選択図】 図1

【書類名】 職権訂正データ
【訂正書類】 特許願

<認定情報・付加情報>

【特許出願人】

【識別番号】 000005821

【住所又は居所】 大阪府門真市大字門真1006番地

【氏名又は名称】 松下電器産業株式会社

【代理人】 申請人

【識別番号】 100090446

【住所又は居所】 大阪府大阪市北区豊崎3丁目2番1号 淀川5番館
5F 中島国際特許事務所

【氏名又は名称】 中島 司朗

出 願 人 履 歴 情 報

識別番号 [000005821]

1. 変更年月日	1990年 8月28日
[変更理由]	新規登録
住 所	大阪府門真市大字門真1006番地
氏 名	松下電器産業株式会社